Computer Vision and Image Analysis Course

Lab 3: Perspective-n-Point projection and Two-View Geometry

Interdisciplinary Space Master (ISM)
Interdisciplinary Centre for Security, Reliability and Trust
Computer Vision, Imaging, and Machine Intelligence (CVI^2) group

Course Responsible: Prof. Djamila Aouada
TA: Dr. Arunkumar Rathinam, Dr. Michele Lynn Jamrozik
arunkumar.rathinam@uni.lu; michele.jamrozik@uni.lu

23 November 2022
Submission Deadline: 30 November 2022

In the previous task, you learnt how to utilize camera intrinsic parameters to do projection from real world to camera frame. In this task you will learn how to estimate Pose information using Perspective-n-point projection and also computation of the 3D coordinates of the points using Triangulation.

3.1 Estimate Pose Information

Compute the pose of the object using the OpenCV SolvePnP

1. Read the **points_1.txt** and **points_2.txt** files. The text file contains the image coordinates of the interesting keypoints in the satellite.

```
# The data is in the format (u,v,visibility)
# u - image coordinate along horizontal axis
# v - along vertical axis
# Visibility - whether the points is visible in the image or not
```

2. Read the **camera.json** for the parameters necessary to construct the camera calibration matrix.

```
# The Camera parameters is in the format below.
nu # number of horizontal[pixels]
nv # number of vertical[pixels]
ppx # horizontal pixel pitch[m / pixel]
ppy # vertical pixel pitch[m / pixel]
fx # focal length[m]
fy # focal length[m]
ccx # center along horizontal [pixels]
ccy # center along vertical[pixels]
dist_coefs # distortion coefficients
```

3. *OpenCV SolvePnP* command will be helpful to find an object pose from 3D-2D point correspondences.

```
import cv2
success, rotation_vector, translation_vector = cv2.solvePnP(objectPoints,
                                                imagePoints, camera_matrix,
                                                dist_coefs, flags)

# objectPoints - Array of object points in the object coordinate space,
# Nx3 where N is the number of points.
# imagePoints - Array of corresponding image points,
# Nx2 where N is the number of points.
# camera_matrix - intrisic matrix of the camera
```

4. Read the **points3d.txt** files for the position of the keypoints (object points in 3D) in the satellite's own reference frame. The keypoint location are in the format (x,y,z).

5. Find the pose of the target in the camera reference frame using SolvePnP. The pose information should be the format position (x,y,z) and quaternion (qx, qy, qz, qw).

6. Investigate different (min. two) PnP refinement options and report whether there is any performance improvement.

3.2 Estimating Fundamental Matrix and Essential Matrix

In a stereo image pair, the fundamental matrix encodes the rigidity constraint of the scene. In this exercise, you are provided with a set of image pairs. Estimate the Fundamental matrix and the Essential matrix.

1. In the folder, there are two images that represent the stereo pair. Read the images using OpenCV.

2. Calculate sift descriptors

```
# calculate sift descriptors
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
sift = cv.SIFT_create()
kp, des = sift.detectAndCompute(gray,None)
```

3. Perform feature matching using BF matcher. Please feel free to use any other feature matcher as well.

```
#feature matching
bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
matches = bf.match(desA,desB)
matches = sorted(matches, key = lambda x:x.distance)
## Create Flann matcher
FLANN_INDEX_KDTREE = 1
flann_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
matcher = cv2.FlannBasedMatcher(flann_params, {})
matches = matcher.knnMatch(desA, desB, 2)
```

4. Extract keypoints from matches

```
for i, (m1,m2) in enumerate(matches):
    if m1.distance < 0.7 * m2.distance:
        ## Notice: How to get the index
        pt1 = kpA[m1.queryIdx].pt
        pt2 = kpB[m1.trainIdx].pt
        KPA.append(pt1)
        KPB.append(pt2)
```

5. Convert keypoints coordinates where the origin is located at the image center

```
# Passes to coordinates centered on center x,y of image
Kp = -1*(kp - (np.array(imageA.shape[0:2])*0.5))
```

6. Estimate Fundamental Matrix using the keypoints

7. Estimate Essential matrix

****END****